



IN THE  
UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): COUTANT et al.

Confirmation No.: 1046

Application No.: 09/702,593

Examiner: Kendall, C.

Filing Date: 10/31/2000

Group Art Unit: 2122

Title: METHOD AND APPARATUS FOR SWITCHING BETWEEN MULTIPLE IMPLEMENTATION  
OF A ROUTINE

Mail Stop Appeal Brief-Patents  
Commissioner For Patents  
PO Box 1450  
Alexandria, VA 22313-1450

**RECEIVED**

JUL 19 2004

Technology Center 2100

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on 05/14/2004.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) \$330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( ) one month	\$110.00
( ) two months	\$420.00
( ) three months	\$950.00
( ) four months	\$1480.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of \$330.00. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: 07/08/04

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number \_\_\_\_\_ on \_\_\_\_\_

Number of pages: 12

Typed Name: Rennae Johnson

Signature: Rennae Johnson

Respectfully submitted,

COUTANT et al.

By LeRoy D. Maunu

LeRoy D. Maunu

Attorney/Agent for Applicant(s)

Reg. No. 35,274

Date: 07/08/04

Telephone No.: (651) 686-6633

09/702,593  
10001275-1


IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant:	Coutant et al.	Examiner:	Kendall
Serial No.:	09/702,593	Group Art Unit:	2122
Filed:	October 31, 2000	Docket No.:	10001275-1 (HPCO.008PA)

Title: METHOD AND APPARATUS FOR SWITCHING BETWEEN  
MULTIPLE IMPLEMENTATIONS OF A ROUTINE

---

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence and the papers, as described hereinabove, are being deposited in the United States Postal Service in triplicate, as first class mail, in an envelope addressed to: Mail Stop Appeal Brief – Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on July 8, 2004.

By:   
Rennae Johnson

APPEAL BRIEF

RECEIVED

JUL 19 2004

Technology Center 2100

Mail Stop Appeal Brief - Patents  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

Sir:

This is an Appeal Brief submitted pursuant to 37 C.F.R. § 1.192 for the above-referenced patent application and is being filed in triplicate.

**I. Real Party in Interest**

The real party in interest is Hewlett-Packard Company having a place of business at 1501 Page Mill Road, Palo Alto, CA. The above referenced patent application is assigned to Hewlett-Packard Company.

**II. Related Appeals and Interferences**

Appellant is unaware of any related appeals or interferences.

07/14/2004 SSESHE1 00000139 082025 09702593  
01 FC:1402 330.00 DA

### **III. Status of Claims**

Claims 1-16 are presented for appeal.

Claims 1-16 stand rejected under 35 USC §102(b) as being anticipated by US patent number 5,276,881 to Chan et al. ("Chan").

The claims presented for appeal are found in the attached Appendix of Appealed Claims.

### **IV. Status of Amendments**

The application was initially filed on October 31, 2000 and included claims 1-16. In reply to a first Office Action, which was mailed on October 29, 2003, a Response was filed on January 23, 2004, and the response traversed the rejection under §102(b). A final Office Action was mailed on April 1, 2004, and a Notice of Appeal was filed on May 14, 2004.

### **V. Summary of Invention**

One embodiment of Appellant's invention is directed to a method for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system. A plurality of implementations of a routine are compiled into respective object code modules (FIG. 1, #108, #114; p. 3, l. 14 – p. 4, l. 21). The routine has an associated name and each implementation is adapted to a selected hardware configuration (FIG. 1, #104; p. 3, ll. 26-27; p. 3, l. 33 - p. 4, l. 16). The object code modules are associated with the name of the routine and respective sets of hardware characteristics (FIG. 2, #152, 154; p. 4, l. 30 – p. 5, l. 3). When the application program is loaded into memory of the computer system, a reference to the routine is resolved using the sets of hardware characteristics and a hardware configuration of the system (FIG. 3, #302, 306; p. 5, ll. 20-30).

### **VI. Issue for Review**

**Is the § 102(b) rejection of claims 1-16 proper when the asserted *Chan* reference fails to teach or suggest every limitation of the claims?**

### **VII. Grouping of Claims**

For purposes of this appeal, claims 1, 2, 13, and 16 are in group I; claims 3, 9, and 14 are in group II; claims 4, 7, 10, and 15 are in group III; and claims 5, 6, 8, 11, and 12 are in group IV. The claims of the different groups do not stand or fall together.

### **VIII. Argument**

**The §102(e) rejection of claims 1, 3-6, 8-11, and 13-17 is improper because the asserted *Chan* reference fails to teach or suggest every limitation of the claims.**

To establish that claims are anticipated, the prior art must show that every limitation in the claims is taught by the cited reference. The *Chan* reference does not teach all the limitations, and therefore, the Office Actions have not established that the claims are anticipated.

The Office Actions fail to show that *Chan* teaches the limitations of the claims 1, 2, 13 and 16 in group I. These limitations include compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration; associating the object code modules with the name of the routine and respective sets of hardware characteristics; and resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system. The rejection fails to show that *Chan* teaches these limitations.

For example, the Office Actions fail to show that *Chan* teaches compiling a plurality of implementations of a routine into respective object code modules. *Chan* does not teach compiling a plurality of implementations. Various portions of *Chan*, as well as the cited teachings, show that *Chan* generates machine-specific modules from a single implementation, not multiple implementations.

*Chan*'s FIG. 2 shows a generation of a single compiler intermediate representation 212 from a single machine independent program 202. The single compiler intermediate representation is used to generate the object code on target computer platforms, and the target computer platforms may not be the same (FIG. 2; col. 9, ll. 59-66). Furthermore, the *Chan* invention is described as "directed to ... distributing a machine independent computer program ... to heterogeneous target computer platforms." (col. 5, ll. 60-64). Therefore, *Chan*'s teachings are clearly directed to a generating respective object code modules from a single implementation, not from a plurality of implementations.

Two sections of *Chan* are cited as teaching the plurality of implementations, col. 60, ll. 13-17 and col. 60, ll. 33-35. Neither of these passages may be reasonably read as compiling a plurality of implementations of a routine into respective object code modules. *Chan*'s col. 60, ll. 13-17 teaches mapping register requirements in the low-level compiler intermediate representation into the register set of the target platform. It is respectfully submitted that this mapping is a sub-function of *Chan*'s ANDF installer, which uses the single compiler intermediate representation (col. 56, ll. 5-11; FIG. 2, #212, #218).

The other cited section portion of *Chan* (col. 60, ll. 30-35) briefly mentions generating object code from input machine instructions and storing the object code in bit patterns expected by the hardware. However, there is no suggestion of compiling a plurality of implementations of a single routine into respective code modules.

The Office Actions further fail to show object code modules being associated with the routine's name and respective sets of hardware characteristics. *Chan* would not need the claimed association since *Chan* generates object code on specific platforms from a compiler intermediate representation (FIG. 2, col. 12, ll. 17-39). The cited portions of *Chan* teach:

...The object file Generator 1358 generates and stores object code in an Object file 1362. The object code is packed into the precise bit patterns expected by the underlying hardware.

The object File Generator 1358 writes other required information to the Object file 1362 according to the format expected by the native linker and loader. This might include Symbol Table 1312 and Type Table 13006 information, relocation information, and object file management information. (col. 60, ll. 30-39).

This portion of *Chan* teaches generating object code for a particular hardware platform. A reasonable reading of associating object code modules with respective sets of hardware characteristics does not read on *Chan*'s teachings. Those skilled in the art would clearly recognize that associating sets of hardware characteristics with object code modules is not taught by only compiling code for a particular platform. Furthermore, if the alleged interpretation of the limitations is accepted, then the "associating" step would be the same as the "compiling" step, and the "associating" step would be unnecessary. Those skilled in the art will appreciate that the associating step has a purpose beyond the installing of object code on a particular platform, as demonstrated in the "resolving" step. *Chan* does not teach these limitations.

The Office Actions also fail to show that *Chan* teaches resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system. Since *Chan* teaches a single object module for a hardware platform, *Chan* would have no apparent need to resolve a reference to a routine using both the hardware configuration of the system and the sets of hardware characteristics associated with the object modules.

*Chan*'s col. 55, ll. 60-65 is cited as teaching these limitations. However, the cited portion of *Chan* teaches:

The HPcode-Plus Object files 1160 deposited into the HPcode-Plus archive file are extracted, installed, and archived into a standard library file at the target computer platform 216, 226. The resulting library file is then searched as usual for resolving references by a native linker on the install site 216, 226. All system specific libraries on the native computer platform 206 used by computer programs 202 must be translated into HPcode-Plus archive files and provided to the target computer platforms 216, 226. (col. 55, ll. 60-65).

It appears that this section of *Chan* refers to installing and linking a library on specific computer platforms. It will be appreciated that the claims relate to resolving references when the application program is loaded into memory, which those skilled in art will recognize as being different from installing. In addition, there appears to be no mention of resolving the references using the associated sets of hardware characteristics. The cited text simply mentions "searching as usual for resolving references by a native linker..."

Regarding claim 2, the limitations include establishing a symbol table having a plurality of entries, each entry including a name of a routine and a reference to an object code module in the library. The Office Action cites *Chan*'s FIG. 13, elements 1308 and 1310, along with *Chan*'s col. 58, ll. 44-53 as teaching these limitations. However, this section of *Chan* teaches using a symbol table for data objects. There is no suggestion of using the symbol table for object modules.

The Office Action fails to establish that *Chan* teaches the limitations of the claims 3, 9, and 14 in group II. For example, claim 3 includes limitations of adding a plurality of entries to the symbol table and associating respective sets of hardware characteristics with the plurality of entries for the routine having a plurality of implementations. None of the cited teachings of *Chan* mention that for a routine having a plurality of implementations, associating respective sets

of hardware characteristics with the plurality of entries in the symbol table. The Office Action cites *Chan's* col. 48, ll. 1-15 as teaching these limitations. However, this text describes creating a symbol table having symbolic identifiers, symbolic information, and symbolic kind information. There is no apparent teaching or even a suggestion of entries in the symbol table being associated with sets of hardware characteristics.

The Office Action fails to establish that *Chan* teaches all the limitations in the claims 4, 7, 10, and 15 of group III. Claim 4 in this group includes limitations of the hardware characteristics including at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system. Even if *Chan* teaches at least one of these hardware characteristics, as explained above, *Chan* does not associate any of these characteristics with information in the symbol table. *Chan* teaches using a machine configuration file to select platform-specific instructions to represent quadruples from the intermediate code (col. 58, ll. 6-35). Therefore, the claims in group III are not shown to be anticipated.

The Office Action fails to establish that *Chan* teaches all the limitations in the claims 5, 6, 8, 11, and 12 of group IV. For example, claim 5 includes limitations of the resolving step further comprising obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware. The cited section of *Chan* does not teach any resolving of a reference to a routine using a configuration file. The Office Action cites *Chan's* col. 58, ll. 18-23 as teaching these limitations. However, this text simply references a machine configuration file containing architecture specific register sizes, co-processor availability, and instruction cycle times. *Chan's* preceding paragraphs explains that this information is used by a low-level code generator to select a sequence of instructions to implement a quadruple from the intermediate code. This clearly does not teach resolving a reference to a routine by using a configuration data file, system identification registers, or system firmware when the program is loaded. Therefore, the claims in group IV are not shown to be anticipated.

**Separate patentability**

Claims 3, 9, and 14 in group II are separately patentable over the claims in the other groups. The claims in group II include limitations of adding a plurality of entries to the symbol table and associating respective sets of hardware characteristics with the plurality of entries for the routine having a plurality of implementations. These limitations are not an obvious extension of any parent claims, nor are these limitations necessarily found in the claims in the other groups (except by dependency). Therefore, claims 3, 9, and 14 in group II are separately patentable over the claims in the other groups.

Claims 4, 7, 10, and 15 in group III are separately patentable over the claims in the other groups. The claims in group III include limitations of the hardware characteristics including at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system. These limitations are not an obvious extension of the claims from which these claims respectively depend, nor are these limitations necessarily found in the claims in the other groups (except by dependency). Furthermore, claim 4 is separately patentable over claim 7 because claim 4 includes the limitations of claims 2 and 3 by dependency, and claim 7 does not, and the limitations of claim 4 are not an obvious extension of the limitations of claim 7. Similarly, claim 4 is separately patentable over claim 10 because the limitations of claim 4 are not an obvious extension of the limitations of claim 10. Therefore, claims 4, 7, 10, and 15 in group III are separately patentable over the claims in the other groups, and claim 4 is also separately patentable over claims 7 and 10.

Claims 5, 6, 8, 11, and 12 in group IV are separately patentable over the claims in the other groups. The claims in group IV include limitations of the resolving step further comprising obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware. These limitations are not an obvious extension of the respective parent claims, nor are these limitations necessarily



09/702,593  
10001275-1

found in the claims in the other groups (except by dependency). Therefore, claims 5, 6, 8, 11, and 12 in group IV are separately patentable over the claims in the other groups.


**IX. Conclusion**

In view of the above, Appellant believes the claimed invention to be patentable. The Office Action makes incomplete and erroneous findings of fact in applying the reference and fails to satisfy the requirements for establishing a *prima facie* case of anticipation. These mistakes are the basis of erroneous conclusions of law pertaining to non-allowability of the claims.

Claims 1-16 remain for consideration. Appellant respectfully requests reversal of the rejections as applied to the appealed claims and allowance of the entire application.

Respectfully submitted,

CRAWFORD MAUNU PLLC  
1270 Northland Drive – Suite 390  
St. Paul, MN 55120  
(651) 686-6633

By:   
Name: LeRoy D. Maunu  
Reg. No.: 35,274

**APPENDIX OF APPEALED CLAIMS (09/702,593)**

1. A computer-implemented method for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system, comprising:
  - compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;
  - associating the object code modules with the name of the routine and respective sets of hardware characteristics; and
  - resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.
2. The method of claim 1, further comprising establishing a symbol table having a plurality of entries, each entry including a name of a routine and a reference to an object code module in the library.
3. The method of claim 2, further comprising, for the routine having a plurality of implementations, adding a plurality of entries to the symbol table and associating respective sets of hardware characteristics with the plurality of entries.
4. The method of claim 3, wherein the hardware characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system.
5. The method of claim 4, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.

6. The method of claim 3, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.
7. The method of claim 1, wherein the hardware characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, and instruction set characteristics.
8. The method of claim 1, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.
9. A computer-implemented method for switching between multiple implementations of a routine in a library of routines that are linked with an application program hosted by a computer system, comprising:
  - establishing a set of hardware configuration characteristics that describe the computer system;
  - establishing a symbol table, the symbol table having one or more entries that include a name of a routine, a set of hardware characteristics, and an address referencing a routine in the library;
  - obtaining a name of a routine having multiple implementations when the library is loaded with the application program into memory of the computer system;
  - matching the name of the routine and the set of hardware configuration characteristics that describe the computer system to an entry in the symbol table; and
  - generating an address in executable code for references to the routine having multiple implementations when the library is loaded with the application program, the address referencing an implementation in the library as identified in the matching step by the entry in the symbol table.

10. The method of claim 9, wherein the hardware configuration characteristics include at least one of clock speed of the processor, processor model, cache configuration of the system, hardware operation latency times, and instruction set characteristics.

11. The method of claim 10, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.

12. The method of claim 9, wherein the resolving step further comprises obtaining the hardware configuration of the system from at least one of a system configuration data file, one or more system identification registers, and system firmware.

13. An apparatus for switching between multiple implementations of a routine in a library of routines that are linked with an application program that is hosted by a computer system, comprising:

means for compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;

means for associating the object code modules with the name of the routine and respective sets of hardware characteristics; and

means for resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.

14. A computer-implemented symbol table for referencing a library of object code modules that implement a plurality of routines, comprising:

a first set of one or more entries, each entry in the first set including a unique name of a routine and a reference to an object code module in the library; and

a second set of one or more entries, each entry in the second set including a shared name of a routine, a set of hardware characteristics, and a reference to an object code module in the library.

15. The symbol table of claim 14, wherein the hardware characteristics include at least one of clock speed of a processor, processor model, cache configuration, hardware operation latency times, instruction set characteristics, bypass characteristics, branch prediction behavior, pre-fetching capability, information describing stall conditions, branch penalties, size and associativity of processor data structures, queue sizes for out-of-order or decoupled processors, and the number of processors in a multi-processor system.

16. A computer program product configured for causing a computer to perform the steps of:  
compiling a plurality of implementations of a routine into respective object code modules, the routine having an associated name and each implementation adapted to a selected hardware configuration;

associating the object code modules with the name of the routine and respective sets of hardware characteristics; and

resolving when the application program is loaded into memory of the computer system, a reference to the routine using the sets of hardware characteristics and a hardware configuration of the system.